LanSpeedTester

Documentazione

Sommario

[1 Introduzione 3](#_Toc59110724)

[1.1 Informazioni sul progetto 3](#_Toc59110725)

[1.2 Abstract 3](#_Toc59110726)

[1.3 Scopo 3](#_Toc59110727)

[2 Analisi 4](#_Toc59110728)

[2.1 Analisi del dominio 4](#_Toc59110729)

[2.2 Analisi e specifica dei requisiti 4](#_Toc59110730)

[2.3 Use case 7](#_Toc59110731)

[2.4 Pianificazione 8](#_Toc59110732)

[2.5 Analisi dei mezzi 8](#_Toc59110733)

[2.5.1 Software 9](#_Toc59110734)

[2.5.2 Hardware 9](#_Toc59110735)

[3 Progettazione 10](#_Toc59110736)

[3.1 Design dell’architettura del sistema 10](#_Toc59110737)

[3.2 Design dei dati 10](#_Toc59110738)

[3.3 Design delle interfacce 10](#_Toc59110739)

[3.4 Design procedurale 11](#_Toc59110740)

[4 Implementazione 11](#_Toc59110741)

[5 Test 13](#_Toc59110742)

[5.1 Protocollo di test 13](#_Toc59110743)

[5.2 Risultati test 16](#_Toc59110744)

[6 Consuntivo 18](#_Toc59110745)

[7 Conclusioni 19](#_Toc59110746)

[7.1 Sviluppi futuri 19](#_Toc59110747)

[7.2 Considerazioni personali 19](#_Toc59110748)

[8 Bibliografia 19](#_Toc59110749)

[8.1 Sitografia 19](#_Toc59110750)

[9 Allegati 19](#_Toc59110751)

# Introduzione

## Informazioni sul progetto

* Mandante: Geo Petrini.
* Partecipanti: Daniele Cereghetti.
* Tempo a disposizione: 1° settembre 2020 – 23 dicembre 2020
* Classe: I3AC.
* Progetto: LanSpeedTester

## Abstract

*L’applicazione permette di verificare la velocità di una rete aziendale dopo un crollo o un rallentamento imputabile a qualsiasi causa. Il programma è utilizzabile principalmente dagli informatici ma è anche possibile un uso da parte di un utente medio. Con questo metodo si velocizza il controllo dell’efficienza della banda perché non si dovranno riaccendere i server e successivamente effettuare uno stress test. Questa soluzione permette di ridurre i tempi di inattività, permettendo agli utenti di ritornare più velocemente al lavoro. Questo applicativo richiede unicamente due computer dotati di Java Runtime Environment, che potranno sviluppare una statistica sui risultati del mini test effettuato.*

## Scopo

Lo scopo del progetto è lo sviluppo di un’applicazione per il test della velocità di trasferimento dati all’interno di una rete locale. Il quale sarà diviso in due parti, la parte del server e la parte del client: il server si occupa di ricevere i dati e ritornarli, invece il client oltre a inviare e ricevere i dati, si preoccupa di cronometrare il tempo che passa tra l’invio e la ricevuta, alla fine del test il client mostra la statistica dei risultati ottenuti con la media dei tempi, il numero di pacchetti ricevuti e persi e la velocità di banda della LAN sulla quale si ha fatto il test.

# Analisi

## Analisi del dominio

Principalmente questo programma verrà utilizzato in ambito aziendale dopo l’avvenimento di un blocco della rete o l’aggiunta di nuove apparecchiature, per verificare che la velocità di banda sia ai livelli richiesti, si può utilizzare questo piccolo applicativo. Attualmente non so come può essere risolto questo problema, ma posso ipotizzare che utilizzino qualche tool per gli stress test.

Per il corretto funzionamento di questo programma non si hanno bisogna di enormi conoscenze informatiche, basta saper cosa è un indirizzo IP, cos’è una porta e cos’è un pacchetto di dati.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: 1** | |
| **Nome** | Software che fa il cambio di ruolo con il server ed il client |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 2** | |
| **Nome** | Il Software non deve essere installato e non salva niente sul PC |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 3** | |
| **Nome** | Utilizza un Linguaggio multi-piattaforma |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Utilizzo il linguaggio Java |

|  |  |
| --- | --- |
| **ID: 4** | |
| **Nome** | L’utente del server sceglie la porta d’ascolto |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 5** | |
| **Nome** | Il server mostra lo stato del client |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 6** | |
| **Nome** | Il server apre porta e accetta connessioni del client |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | questa cosa server anche per l'utilizzo normale, ovvero che per far funzionare correttamente il programma, prima si fa partire il server e successivamente quando il server è "online" si fa "accende il client |

|  |  |
| --- | --- |
| **ID: 7** | |
| **Nome** | Programma da terminale (CLI) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 8** | |
| **Nome** | Server e client possono sfruttare il multithreading |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il server deve accettare una o più connessione in entrata, il client deve poter aprire più connessioni verso lo stesso server; entrambi sulla stessa porta o su porte diverse |

|  |  |
| --- | --- |
| **ID: 9** | |
| **Nome** | L’utente del client deve scegliere il peso di ogni pacchetto da inviare al server |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

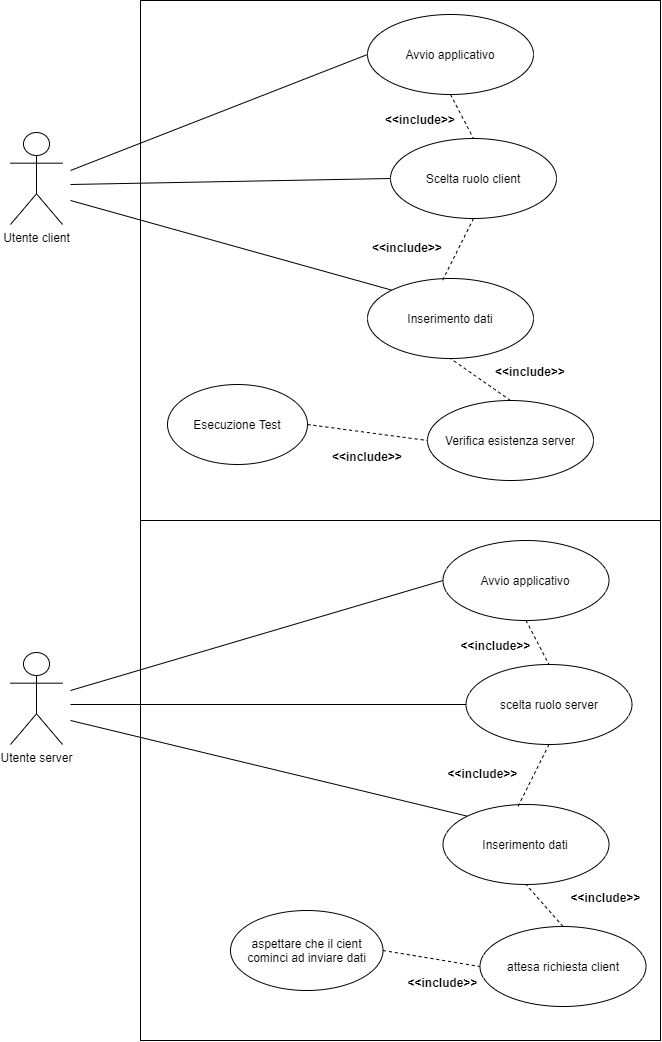
|  |  |
| --- | --- |
| **ID: 10** | |
| **Nome** | L’utente del client deve scegliere il numero di connessioni da fare al server |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 11** | |
| **Nome** | Il client deve mostrare la statistica dei risultati |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 12** | |
| **Nome** | Misurazione dei tempi al millisecondo |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

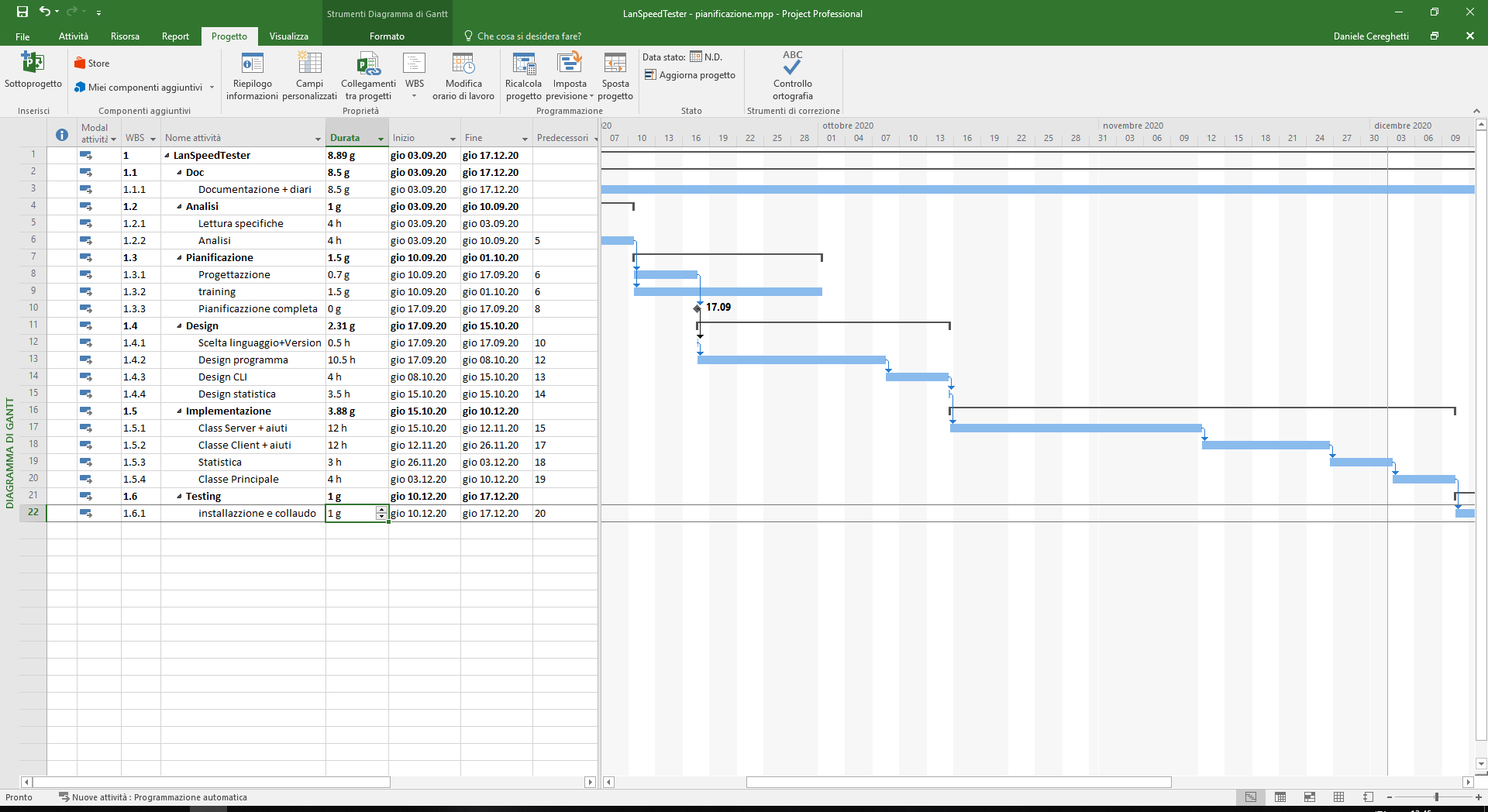
|  |  |
| --- | --- |
| **ID: 13** | |
| **Nome** | Riportare nelle statistiche se è stato sfruttato il multithreading o no |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** |  |

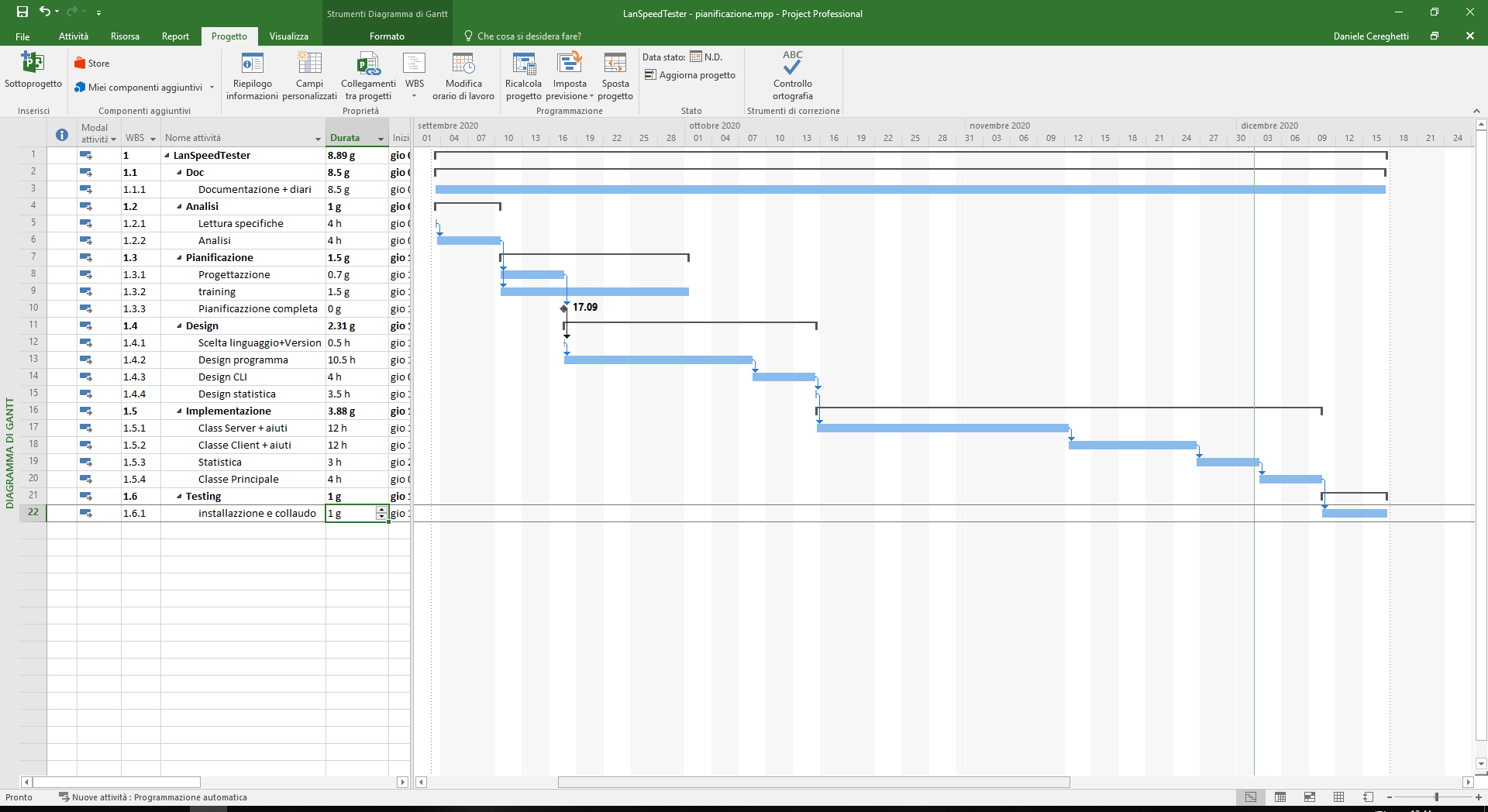
## Use case



L’unica cosa che hanno in comune i due utenti è il fatto che l’applicativo va avviato, poi l’utente server sceglie il ruolo di server, poi inserisce la porta sulla quale il server andrà ad ascoltare. Successivamente il client sceglie il ruolo di client, immette i dati per il test, l’IP del server e la porta. Da quel punto in poi gli utenti possono solo stare a guardare i alcuni parametri, e nel caso del client a fine test vedrà comparire una statistica.

## Pianificazione





## Analisi dei mezzi

Una macchina con cui scrivere un progetto, una macchina virtuale per fare i test e una connessione a internet per le ricerche sul linguaggio, disegno architettura, utilizzo di GitHub, …

### Software

Ide per Java:

* NetBeans IDE 8.2 o superiore

Versione di java (JDK):

* java version "1.8.0\_172" o superiore

Gantt:

* Microsoft Project 2016

Use case, design:

* Microsoft Visio 2016
* <https://app.diagrams.net/>

### Hardware

* CPU 🡪 intel core i7-7700 @ 3.6 GHz
* RAM 🡪 16 GB
* SSD 🡪 500 GB
* Architettura 🡪 64 bit

# Progettazione

## Design dell’architettura del sistema

Ecco lo schema di come funziona il programma:

Invio dati client per il test



Invio dati server per il test



Ci inviamo sequenze di byte a vicenda per il test



Prima che accada quello riportato nello schema sopra, l’applicativo va avviato su entrambe le macchine, entrambi scelgono il ruolo a loro destinato. L’utente server sceglie la porta d’ascolto, successivamente il client inserisce i dati per il test, l’IP del server e la porta. Poi cominciano a scambiarsi informazioni e se va tutto a buon fine il test ha inizio.

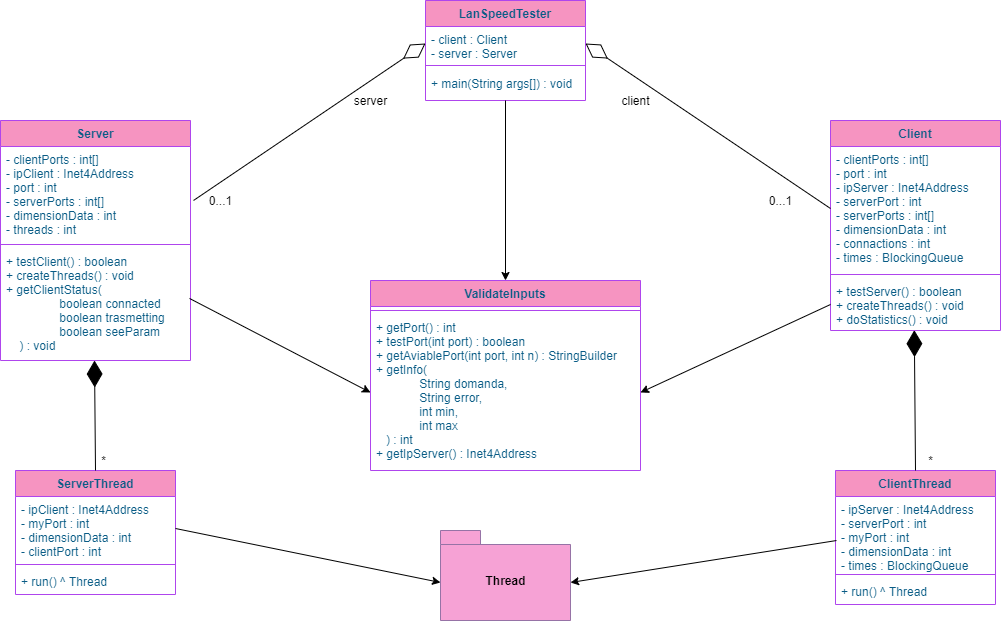
## Design dei dati

Come riportato qua sopra ci sono 3 fasi, la prima e la seconda fase utilizzano una connessione TCP, mentre la terza usa il protocollo UDP. Nella prima e nella seconda vengono inviate delle informazioni necessarie al test, essendo indispensabili ho utilizzato il TCP, invece nell’ultima fase ho scelto l’UDP perché in un test di velocità si ha bisogno di tutta la velocità possibile, e il protocollo UDP è molto più rapido rispetto al TCP e anche perché i dati che andrebbero persi non sono importanti.

## Design delle interfacce

Questo programma sfrutta il multithreading, infatti sono loro che si occupano di fare i test. Sia il client che il server creano degli array di byte che saranno vuoti (contengono solo tanti 0), successivamente aprono un socket verso il dispositivo assegnato, per poi iniziare a trasmettere dati. Nel caso del client ci sono alcune differenze, lui cronometra il tempo di invio e di ritorno e ne fa una media, poi la media viene salvata in una lista che successivamente verrà utilizzata per fare la statistica.

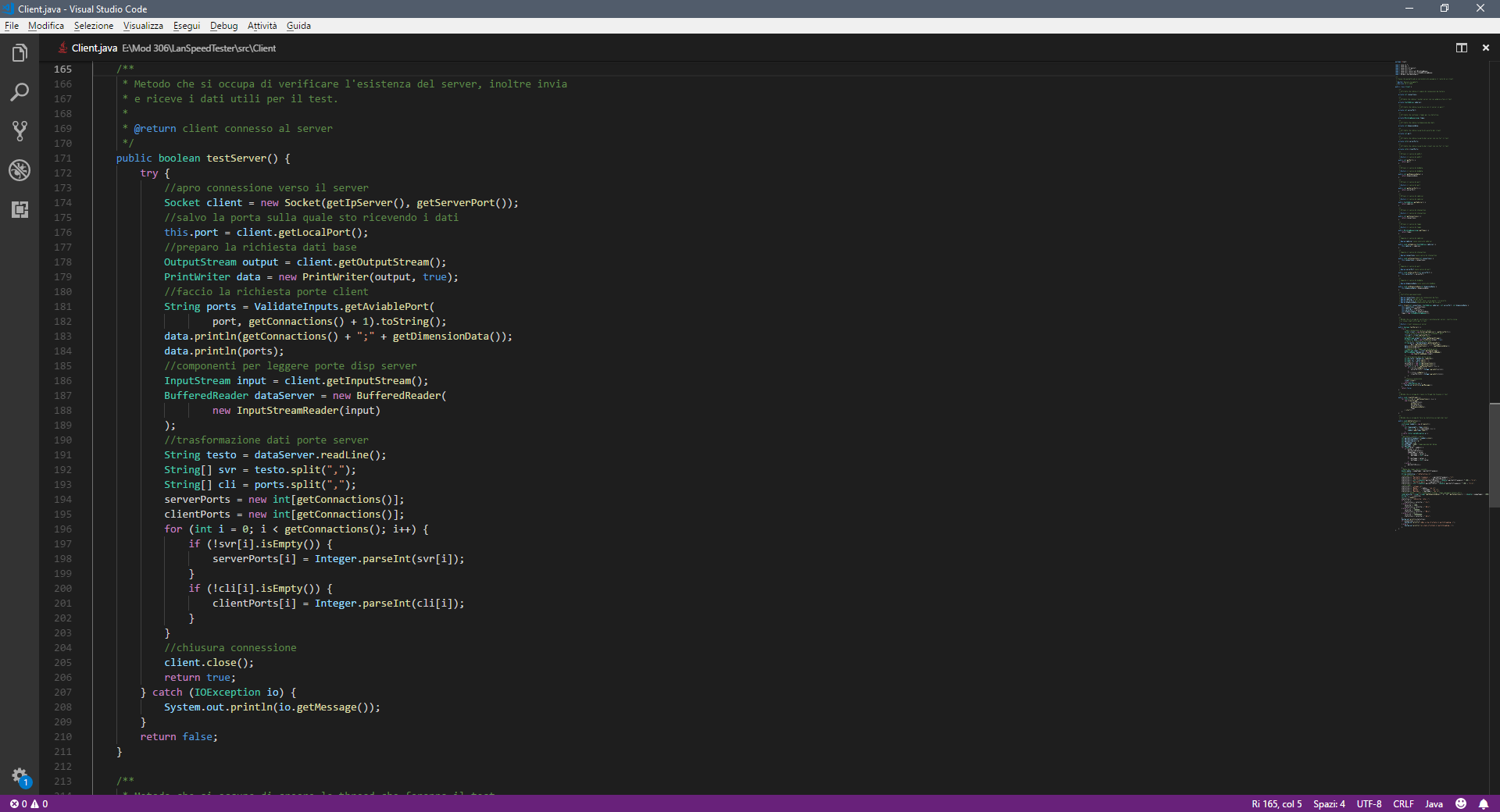
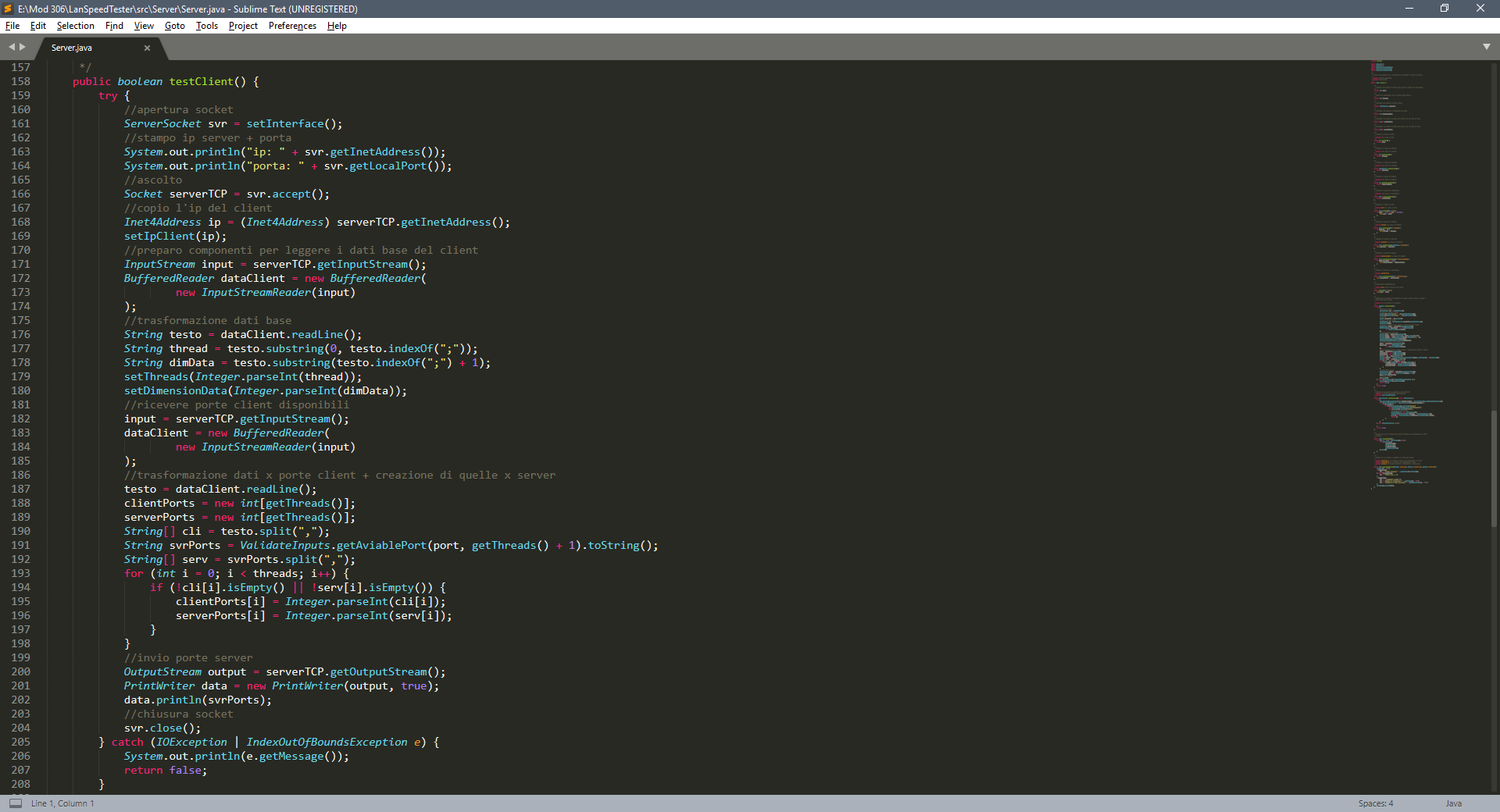
## Design procedurale



# Implementazione

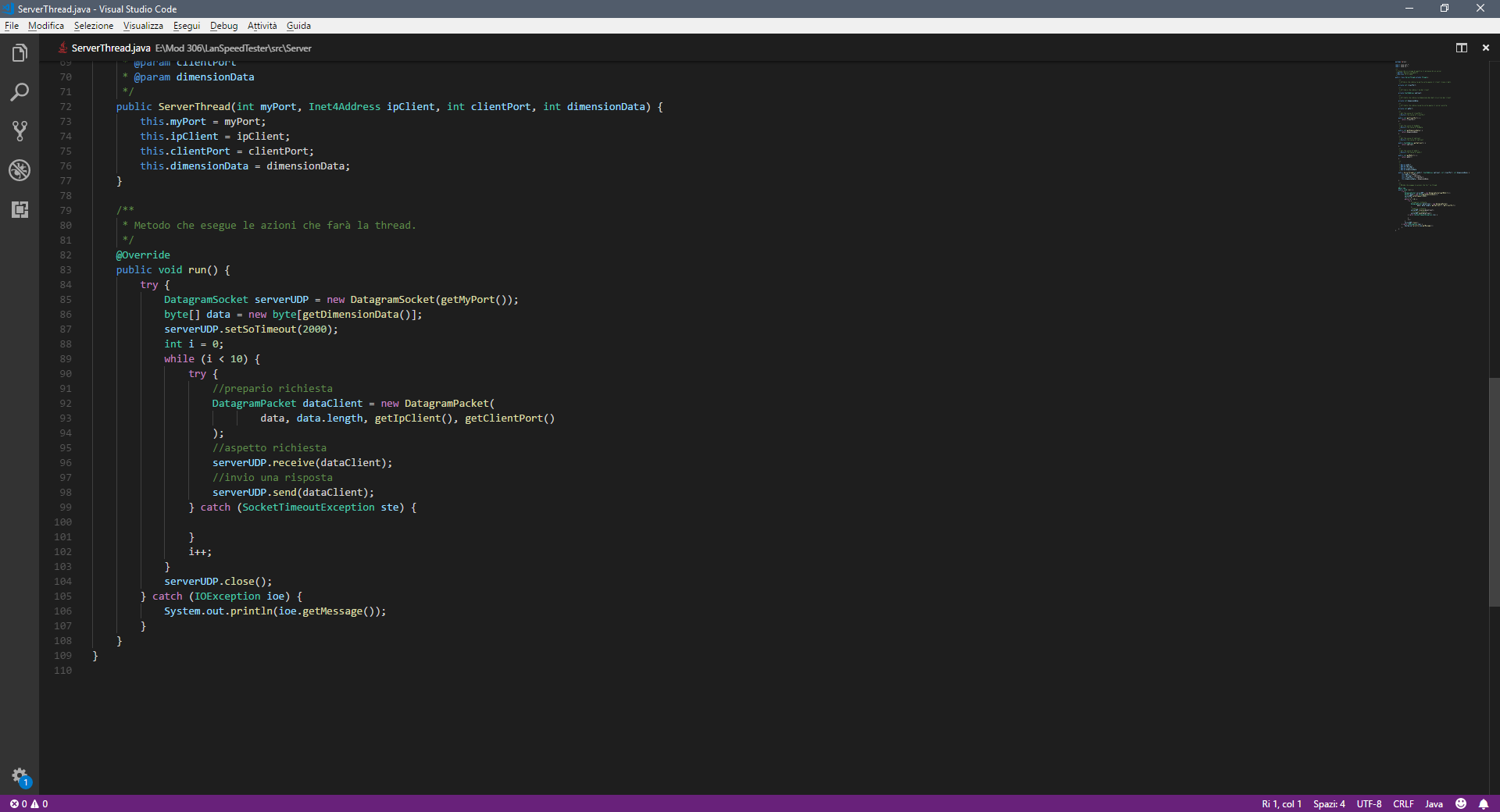
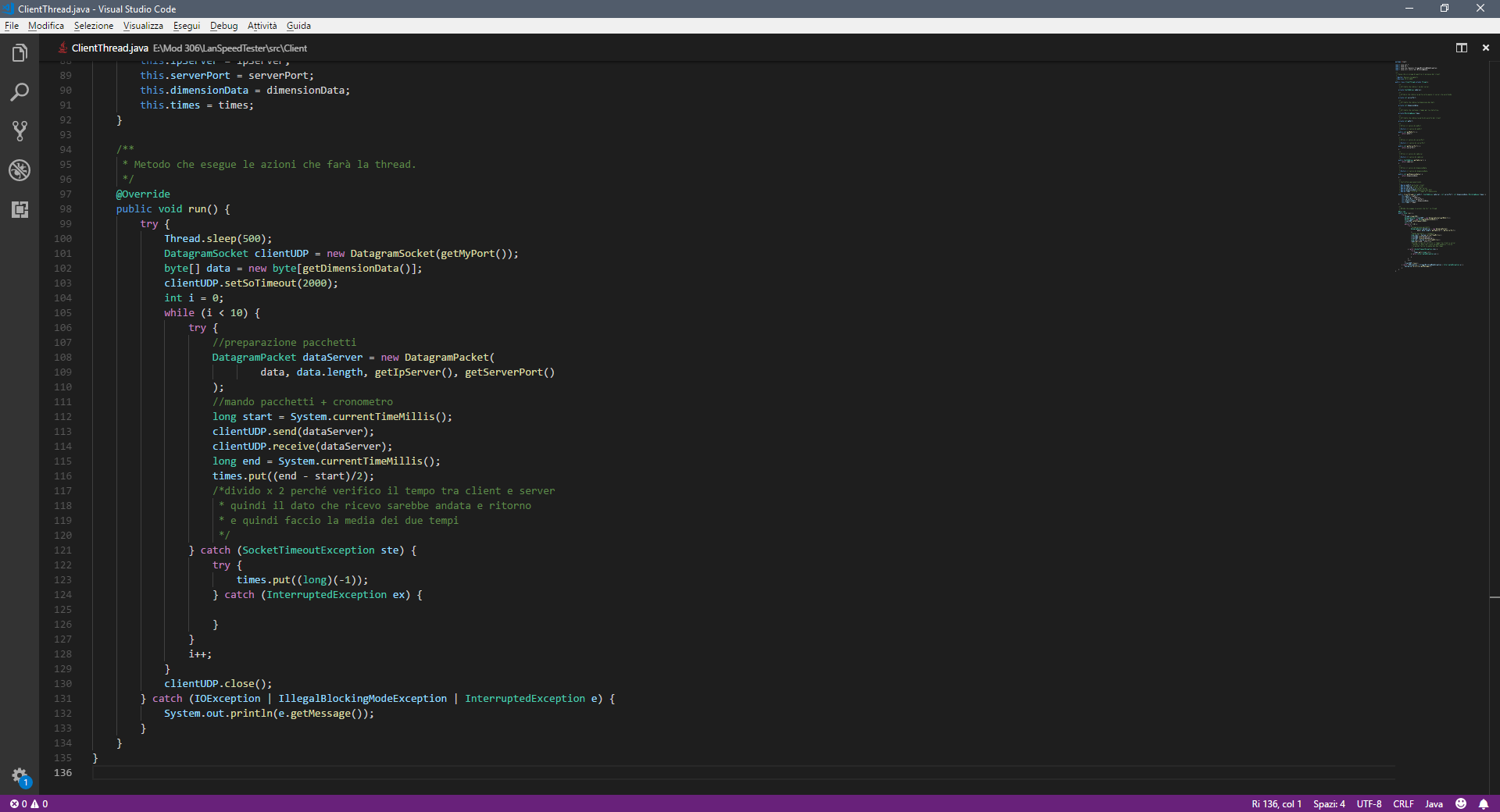
Non ho avuto parecchie difficoltà o parti critiche a fare questo progetto, però quei pochi problemi che sono venuti a galla hanno avuto una complessità che per me non sono così facili.

Le seguenti immagini mostrano a parer mio la parte più complicata del programma, ovvero le due classi che si occupano della comunicazione iniziale, ma più precisamente quando le due macchine devono cominciare a parlarsi, infatti solamente in questi due punti sono presenti dei commenti all’interno del metodo visto che all’inizio ho avuto parecchi problemi in questo punto.

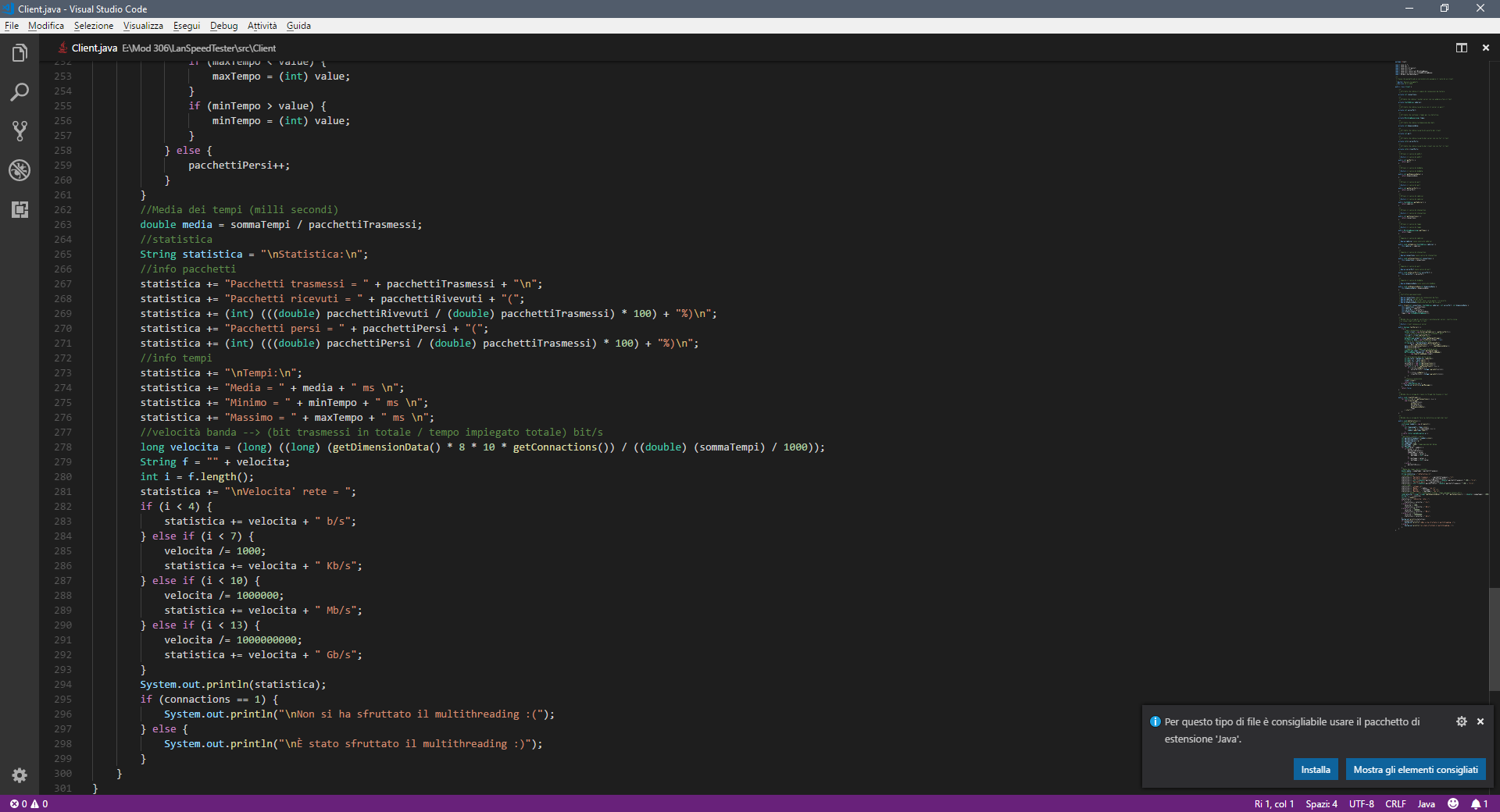
Classe: Client Classe: Server

Ma non è stato solo quel punto a darmi problemi, anche le due classi che si occupano di fare il test hanno richiesto il loro tempo, e hanno avuto anche un cambio di progettazione, prima lavoravano su una porta sola, ma per colpa di molteplici errori ho preferito cambiare progettazione (vedi diario[[1]](#footnote-1) del 22-10-2020), per fortuna questo cambio non mi ha portato via troppo tempo, infatti dopo aver cambiato ho avuto problemi nell’assegnamento delle porte (porte del server e del client). Ripeto l’esecuzione questa esecuzione (invio 🡪 ricevo e viceversa) 10 volte perché nel caso ci fosse una sola connessione il test ha più dati a disposizione e la statistica finale risulta più coerente con la realtà.

Classe: ClientThread Classe: ServerThread

Un altro problema è stato il nel calcolo della velocità della rete, perché ogni tanto usciva l’errore della divisione per zero oppure il risultato che usciva non era corretto, alla fine facendo il cast a double sui tempi sono riuscito a sistemare questo problema.



Concludo questo punto dicendo che il risultato di questo progetto rispetta tutte le caratteristiche della progettazione, quindi il progetto può essere concluso con successo.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 1  1 | **Descrizione:** | Software che fa il cambio di ruolo con il server ed il client |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. Aprire il terminale, andare sotto la cartella nella quale si trova il programma ed eseguire il programma (java LanSpeedTester) 2. Inserire il numero che corrisponderà al ruolo del server (1) oppure il client (2) | | |
| **Risultati attesi:** | Caso 1 (server):   * Vengono richiesti i parametri per il server   Caso 2 (client):   * Vengono richiesti i parametri per il client | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 2  2 | **Descrizione:** | Il Software non deve essere installato e non salva niente sul PC |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | * + - 1. Aprire il terminale, andare sotto la cartella nella quale si trova il programma ed eseguire il programma (java LanSpeedTester)       2. Aspettare che l’esecuzione del programma finisca | | |
| **Risultati attesi:** | All’avvio non deve mostrare nessuna richiesta di installazione.  Al suo termine non crei, modifichi e/o elimini nessun file dal PC | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 3  3 | **Descrizione:** | Utilizza un Linguaggio multi-piattaforma |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | Installare una versione di java compatibile con il programma | | |
| **Risultati attesi:** | Che alla sua partenza su un sistema operativo diverso da Windows, non dia errori e funzioni come su Windows. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 4  4 | **Descrizione:** | L’utente del server sceglie la porta d’ascolto |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. scegliere il ruolo di server con il numero 1 all’avvio del programma 2. scegliere una porta d’ascolto | | |
| **Risultati attesi:** | Accettazione della porta e apertura del socket del server | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 5  5 | **Descrizione:** | Il server mostra lo stato del client |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | * + - 1. scegliere il ruolo di server       2. scegliere la porta d’ascolto | | |
| **Risultati attesi:** | Mostra se il client è connesso oppure no, e quando un client si connette viene mostrato che è connesso e mostra le impostazioni del test | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 6  6 | **Descrizione:** | Il server apre porta e accetta connessioni del client |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | scegliere ruolo server  scegliere porta ascolto | | |
| **Risultati attesi:** | Mostri che il server stia effettivamente ascoltando, facendo visualizzare l’ip del server e la sua porta. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 7  7 | **Descrizione:** | Programma da terminale (CLI) |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | eseguire il programma | | |
| **Risultati attesi:** | Non mostri nessun tipo di frame ma che vengano visualizzate le cose nel terminale | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 8  8 | **Descrizione:** | Server e client possono sfruttare il multithreading |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. scelgo ruolo client 2. scelgo il numero di connessioni da testare: tra 1 e 200 | | |
| **Risultati attesi:** | Il server deve accettare una o più connessione in entrata, il client deve poter aprire più connessioni verso lo stesso server; entrambi su porte diverse. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 9  9 | **Descrizione:** | L’utente del client deve scegliere il peso di ogni pacchetto da inviare al server |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. scegliere ruolo client 2. scegliere il numero di connessioni 3. scegliere il peso di ogni pacchetto | | |
| **Risultati attesi:** | Accetti il numero inserito e che successivamente venga visualizzato sul server | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 10  10 | **Descrizione:** | L’utente del client deve scegliere il numero di connessioni da effettuare al server |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. scegliere ruolo client 2. scegliere il numero di connessioni | | |
| **Risultati attesi:** | Accetti il numero inserito e che successivamente venga visualizzato sul server | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 11  11 | **Descrizione:** | Il client deve mostrare la statistica dei risultati |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. scegliere ruolo client 2. effettuare tutti i passaggi per il test che verranno chiesti 3. aspettare che il programma termini | | |
| **Risultati attesi:** | Prima del termine del programma venga mostrata la statistica sul test effettuato | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 12  12 | **Descrizione:** | Misurazione dei tempi in millisecondi |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. scegliere ruolo client 2. effettuare tutti i passaggi per il test che verranno chiesti 3. aspettare che il programma termini | | |
| **Risultati attesi:** | Nella test i tempi siano misurati in millisecondi e che nella statistica i tempi siano mostrati in millisecondi. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 13  13 | **Descrizione:** | Riportare nelle statistiche se è stato sfruttato il multithreading o no |
| **Prerequisiti:** | Nessuno | | |
| **Procedura:** | 1. scegliere ruolo client 2. effettuare tutti i passaggi per il test che verranno chiesti 3. aspettare che il programma termini | | |
| **Risultati attesi:** | Alla fine della statistica venga mostrato se si ha sfruttato il multithreading oppure no | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 1  1 | **Descrizione:** | Software che fa il cambio di ruolo con il server ed il client |
| **Risultati:** | Caso 1 (server):   * Vengono richiesti i parametri per il server   Caso 2 (client):   * Vengono richiesti i parametri per il client | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 2  2 | **Descrizione:** | Il Software non deve essere installato e non salva niente sul PC |
| **Risultati:** | nessuna richiesta di installazione e nessun file creato, modificato o cancellato | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 3  3 | **Descrizione:** | Utilizza un Linguaggio multi-piattaforma |
| **Risultati:** | funziona anche su OS diversi | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 4  4 | **Descrizione:** | L’utente del server sceglie la porta d’ascolto |
| **Risultati:** | Accetta la porta e apre il socket del server | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 5  5 | **Descrizione:** | Il server mostra lo stato del client |
| **Risultati:** | Mostra che il client non è connesso, quando si connette il client viene mostrato questo evento | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 6  6 | **Descrizione:** | Il server apre porta e accetta connessioni del client |
| **Risultati:** | Mostra l’ip del server e la porta sulla quale è in ascolto | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 7  7 | **Descrizione:** | Programma da terminale (CLI) |
| **Risultati:** | Non si apre nessun tipo di Frame e l’esecuzione viene eseguita sul terminale | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 8  8 | **Descrizione:** | Server e client possono sfruttare il multithreading |
| **Risultati:** | Il server accetta una o più connessione in entrata, il client apre più connessioni verso lo stesso punto; entrambi su porte diverse. | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 9  9 | **Descrizione:** | L’utente del client deve scegliere il peso di ogni pacchetto da inviare al server |
| **Risultati:** | Accetta il numero inserito e che successivamente venga visualizzato sul server | | |
| **Risultato finale:** | Positivo | | |

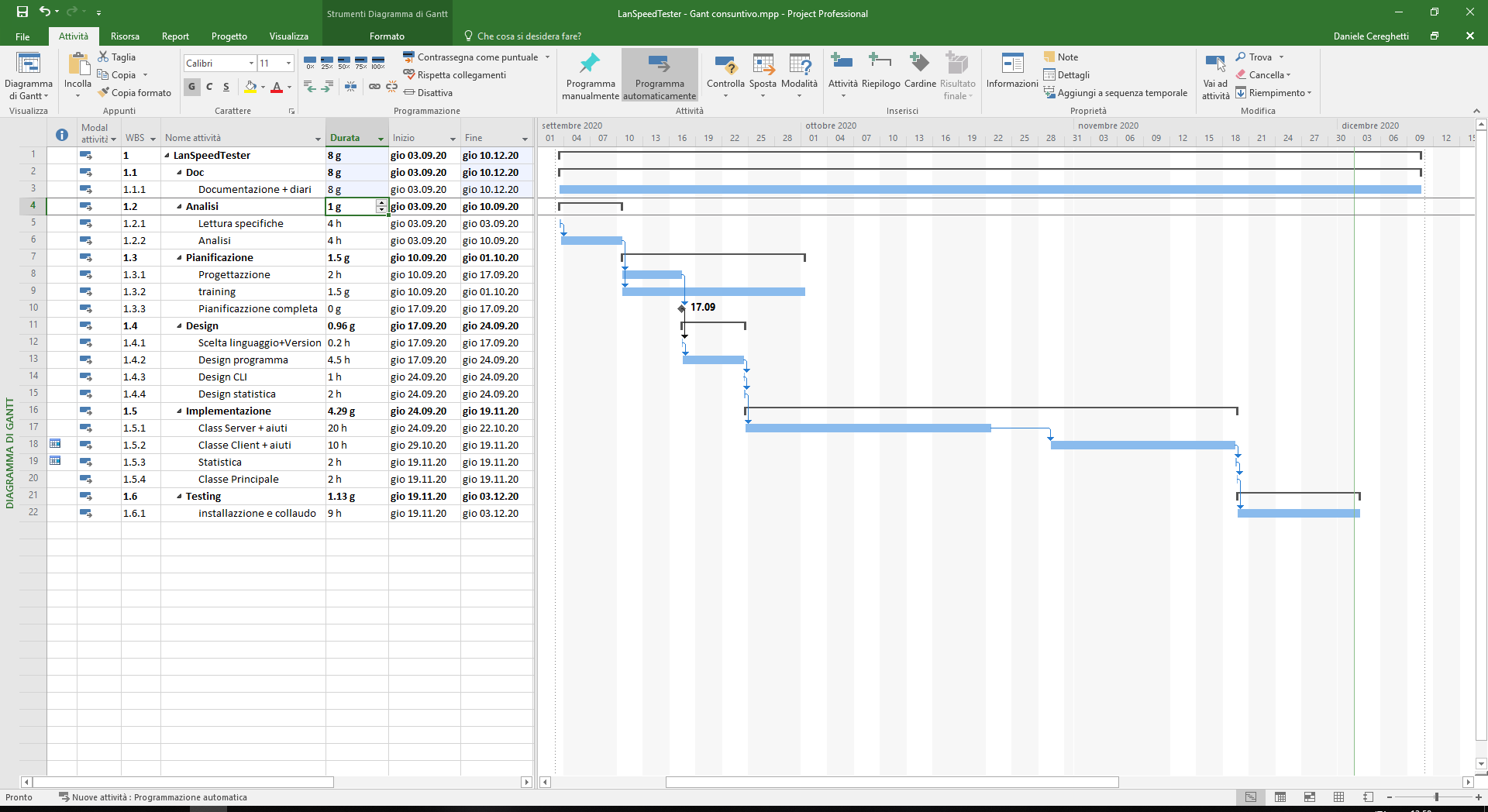
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 10  10 | **Descrizione:** | L’utente del client deve scegliere il numero di connessioni da effettuare al server |
| **Risultati:** | Accetta il numero inserito e che successivamente venga visualizzato sul server | | |
| **Risultato finale:** | Positivo | | |

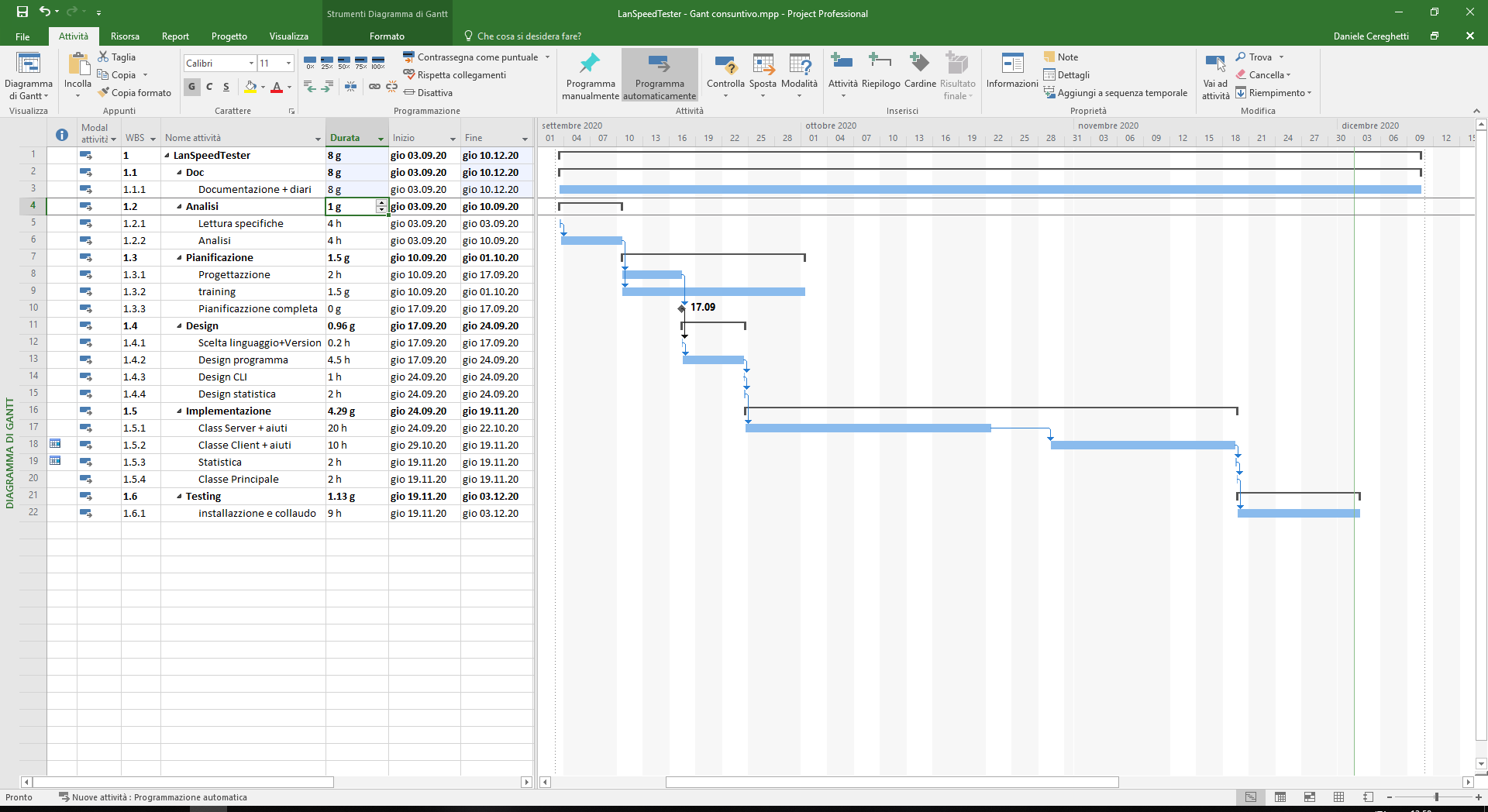
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 11  11 | **Descrizione:** | Il client deve mostrare la statistica dei risultati |
| **Risultati:** | Mostra la statistica prima che termini il programma | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 12  12 | **Descrizione:** | Misurazione dei tempi in millisecondi |
| **Risultati:** | I tempi sono calcolati e visualizzati in millisecondi. | | |
| **Risultato finale:** | Positivo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | 13  13 | **Descrizione:** | Riportare nelle statistiche se è stato sfruttato il multithreading o no |
| **Risultati:** | Alla fine della statistica è mostrato se si ha sfruttato il multithreading oppure no | | |
| **Risultato finale:** | Positivo | | |

# Consuntivo





Confrontando il gantt consuntivo con quello di pianificazione si nota che fino alla pianificazione sono stato al passo con la pianificazione, ma non capisco perché il design è durato molto meno di quello che avevo pensato, e mi ha anche permesso di avere un certo anticipo. Ma nonostante avessi preso un certo vantaggio pensavo che fosse stato più facile scrivere il codice, invece è durato poco di più ma ho avuto parecchi problemi per la parte server, il client invece resta nel suo tempo ideale, invece la cosa che mi ha stupito in positivo è stata la velocità con cui ho fatto la statistica e la classe principale. La parte di testing invece è durata più di quello previsto perché alcune formule davano gli errori e ci ho messo un po’ a trovare una soluzione.

# Conclusioni

## Sviluppi futuri

Si potrebbe far supportare l’IPv6, migliorare il metodo di invio dei dati perché adesso in RAM si allocano (nei casi estremi) tra i 100 e i 130 MB, quindi si potrebbe memorizzare un byte solo, e inviarlo N volte così da inviare in totale il numero di byte richiesti dall’utente.

## Considerazioni personali

Questo progetto mi ha entusiasmato, ho imparato cose nuove e ha messo alla prova le mie conoscenze e devo dire che sono contento di quello che ho fatto, inoltre mi ha aiutato a capire che per capire e fare la logica e ho bisogno di studiarmela a fondo e anche di tempo.

# Bibliografia

## Sitografia

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/package-summary.html>

*Java.net (java SE 12 & JDK 12)*, 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/Socket.html>

*Socket (java SE 12 & JDK 12)*, 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/ServerSocket.html>

*ServerSocket (java SE 12 & JDK 12)*, 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/DatagramSocket.html>

*DatagramSocket (java SE 12 & JDK 12)*, 15.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/DatagramPacket.html>

*DatagramPacket (java SE 12 & JDK 12)*, 15.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/lang/Thread.html>

*Thread (java SE 12 & JDK 12)*, 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/Inet4Address.html>

*Inet4Address (java SE 12 & JDK 12)*, 8.10.2020

1. <https://stackoverflow.com/questions/6916398/communicating-between-two-threads>

*Forum per Queue Java*, 29.10.2020

1. <https://www.codejava.net/java-se/networking/java-socket-server-examples-tcp-ip>

*Esempio socket tcp*, 22.10.2020

1. <https://www.tutorialspoint.com/javaexamples/net_multisoc.htm>

*Esempio socket tcp*, 22.10.2020

1. <https://www.codejava.net/java-se/networking/java-udp-client-server-program-example>

*Esempio socket udp*, 15.10.2020

# Allegati

Per i diari, codici sorgente e prodotto finale 🡪 <https://github.com/Daniele-Cereghetti/LanSpeedTester>

1. Link: <https://github.com/Daniele-Cereghetti/LanSpeedTester/tree/master/Diario> [↑](#footnote-ref-1)